

AD-A104 179

BEDFORD RESEARCH ASSOCIATES MA
CROSS CORRELATIONS FOR TWO-DIMENSIONAL GEOSYNCHRONOUS SATELLITE--EIC(U)
APR 80 P TSIPOURAS, T COSTELLO
UNCLASSIFIED SCIENTIFIC-4

F/6 5/8

F19628-78-C-0083

NL

AFGL-TR-81-0061

1 of 1
AD-A104 179

END
DATE
FILED
10-81
DTIC

11

AFGL-TR-81-0061

163-111

(2)

CROSS CORRELATIONS FOR TWO-DIMENSIONAL
GEOSYNCHRONOUS SATELLITE IMAGERY DATA

LEVEL II

AD A104179

P. Tsipouras
T. Costello

Bedford Research Associates
2 DeAngelo Drive
Bedford, Massachusetts 01730

Scientific Report No. 4

1 April, 1980

DTIC
ELECTED
SEP 14 1981
H

Approved for public release; distribution unlimited

AIR FORCE GEOPHYSICS LABORATORY
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
HANSCOM AFB, MASSACHUSETTS 01731

DMC FILE COPY

830 14100

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER <i>(18) AFGL-TR-81-0061</i>	2. GOVT ACCESSION NO. <i>AD-A204 179</i>	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) <i>(6) Cross Correlations for Two-Dimensional Geosynchronous Satellite Imagery Data</i>	5. TYPE OF REPORT & PERIOD COVERED <i>(14) Scientific Report No. 4</i>	
7. AUTHOR(s) <i>(10) P. Tsipouras T. Costello</i>	6. PERFORMING ORG. REPORT NUMBER <i>(15) F19628-78-C-0083</i>	
9. PERFORMING ORGANIZATION NAME AND ADDRESS <i>Bedford Research Associates 2 DeAngelo Drive Bedford, MA. 01730</i>	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS <i>(16) 9993XXXX (17) XX</i>	
11. CONTROLLING OFFICE NAME AND ADDRESS <i>Air Force Geophysics Laboratory Hanscom AFB, MA. 01731, Monitor/Paul Tsipouras SUWA</i>	12. REPORT DATE <i>(11) 1 April, 1980</i>	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	13. NUMBER OF PAGES <i>56 (12) 49</i>	
16. DISTRIBUTION STATEMENT (of this Report)	15. SECURITY CLASS. (of this report) <i>Unclassified</i>	
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
18. SUPPLEMENTARY NOTES <i>* Air Force Geophysics Laboratories Analysis and Simulation Branch (SUWA) Hanscom AFB, MA. 01731</i>	19. KEY WORDS (Continue on reverse side if necessary and identify by block number) <i>Geosynchronous satellite images, Two-Dimensional FFT Cross Correlation.</i>	
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) <i>The usage of the FFT and other fast unitary transforms for the determination of cross-correlations of geosynchronous satellite images is investigated. The basic background information on the techniques is summarized and a computer program embodying the two-dimensional FFT approach adopted is presented and its usage discussed.</i>		

TABLE OF CONTENTS

Section One	•	•	Summary	5
Section Two	•	•	Fourier Transform Methods	9
Section Three	•	•	Other Transform Techniques	27
Section Four	•	•	References	41
Section Five	•	•	Program Description and Use	45

Accession Per	
MISS	STAN
DATA TAB	<input checked="" type="checkbox"/>
UNIVERSITY	<input type="checkbox"/>
Scanning	<input type="checkbox"/>
No.	
Distribution	
Availability	
1	2
3	4
5	6
7	8
9	10
11	12
13	14
15	16
17	18
19	20
21	22
23	24
25	26
27	28
29	30
31	32
33	34
35	36
37	38
39	40
41	42
43	44
45	46
47	48
49	50
51	52
53	54
55	56
57	58
59	60
61	62
63	64
65	66
67	68
69	70
71	72
73	74
75	76
77	78
79	80
81	82
83	84
85	86
87	88
89	90
91	92
93	94
95	96
97	98
99	100

SECTION I

SUMMARY

4-Blank

SUMMARY

Problem 4006 - Cross Correlation Matrix by FFT

The problem aim was to produce a computer program for determining cross correlations of geosynchronous satellite images by means of a two-dimensional Fast Fourier Transform. Features to be addressed in deciding on the coding for the program were accuracy and speed.

A computer program has been written which reads data from a permanent file, sets up arrays of specified size, and at the user's option smooths the edges of the tw-dimensional array via a \cos^2 weighting method. The program then uses our two-dimensional FFT algorithm to compute the cross correlation matrix for the two arrays. An option is available which searches by row and column to locate the maximum entry in the row/column in the hope that this information will indicate the width of the peaks.

The algorithm used in our routine was bench-marked against three commonly used FFT's. In the most competitive case, our routine was approximately 5 times faster; in the least competitive, we were approximately 20 times faster.

Test data were generated and run. The output was fully consistent with theoretical analyses. The test case corresponded to a two-dimensional square wave. This was chosen for its simplicity of structure, while at the same time providing a solid check on the accuracy and working order of the algorithm.

While the computer program was being developed, alternative approaches were investigated. Specific attention was given to methods which would portentially increase processing speed. The contemporary journals on information and image processing contain an abundance of references to Walsh, Haar and other orthogonal families of functions. The great strength

6-Blank

of the Walsh and Haar transforms is the decrease in processing times by factors of 2-20 over comparable times using the FFT algorithm. After detailed study it was decided that these methods could not profitably be applied to this problem at this time. The principal drawback is the state of the art in interpreting and decoding the convolution and deconvolution via the discrete fast Haar and fast Walsh transforms. Furthermore, since the transform variable space does not have the same "intuitively pleasing" relationship as do time and frequency (Fourier transform pairs), it is not at all obvious which way the interpretations should (or could) be pursued.

S E C T I O N I I

FOURIER TRANSFORM METHODS

Let $f(x)$ be a continuous function of a real variable x . The Fourier transform of $f(x)$, denoted by $\mathcal{F}\{f(x)\}$, is defined by the equation

$$\mathcal{F}\{f(x)\} = F(u) \int_{-\infty}^{\infty} f(x) \exp \left[-j 2\pi u x \right] dx \quad (1-1)$$

where $j = \sqrt{-1}$.

Given $F(u)$, $f(x)$ can be obtained by using the inverse Fourier transform

$$\begin{aligned} \mathcal{F}^{-1}\{F(u)\} &= f(x) \\ &= \int_{-\infty}^{\infty} F(u) \exp \left[j 2\pi u x \right] du \end{aligned} \quad (1-2)$$

Equations (1-1) and (1-2), which are called the Fourier transform pair, can be shown to exist if $f(x)$ is continuous and integrable and $F(u)$ is integrable. These conditions are almost always satisfied in practice.

We will be concerned with functions $f(x)$ which are real. The Fourier transform of a real function, however, is generally complex; that is,

$$F(u) = R(u) + jI(u) \quad (1-3)$$

where $R(u)$ and $I(u)$ are, respectively, the real and imaginary components of $F(u)$. It is often convenient to express Eq. (1-3) in exponential form:

$$F(u) = |F(u)| e^{j\Phi(u)} \quad (1-4)$$

where $|F(u)| = \left[R^2(u) + I^2(u) \right]^{1/2}$ (1-5)

and $\Phi(u) = \tan^{-1} \left[\frac{I(u)}{R(u)} \right]$ (1-6)

The magnitude function $F(u)$ is called the Fourier spectrum of $f(x)$, and $\phi(u)$ its phase angle. The square of the spectrum

$$E(u) = |F(u)|^2 = R^2(u) + I^2(u) \quad (1-7)$$

is commonly referred to as the energy spectrum of $f(x)$.

The variable u appearing in the Fourier transform is often called the frequency variable. This name arises from the fact that, using Euler's formula, the exponential term, $\exp[-j2\pi ux]$, may be expressed in the form:

$$\exp[-2\pi ux] = \cos 2\pi ux - j \sin 2\pi ux \quad (1-8)$$

If we interpret the integral in Eq. (1-1) as a limit-summation of discrete terms, it is evident that $F(u)$ is composed of an infinite sum of sine and cosine terms, and that each value of u determines the frequency of its corresponding sine-cosine pair.

For the step function shown in Fig 1 its Fourier transform is obtained from Eq. (1-1) as follows:

$$\begin{aligned} F(u) &= \int_{-\infty}^{\infty} f(x) \exp[-j2\pi ux] dx \\ &= \int_0^X A \exp[-j2\pi ux] dx \\ &= \frac{A}{\pi u} \sin(\pi u X) e^{-j\pi u X} \end{aligned}$$

which is a complex function.

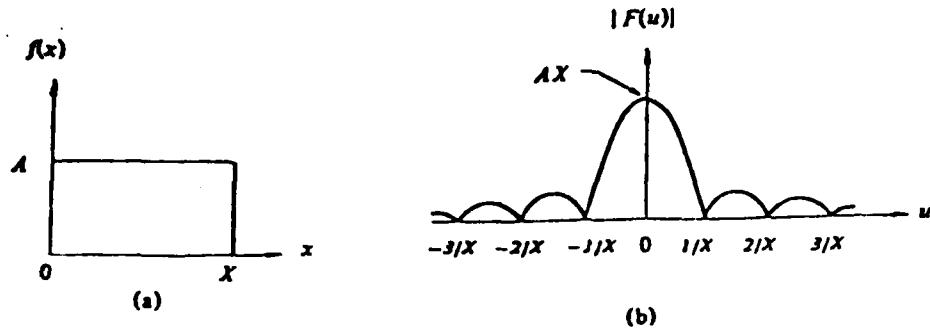


Figure 1. A step function and its Fourier spectrum.

The Fourier transform can be easily extended to a function $f(x,y)$ of two variables. If $f(x,y)$ is continuous and integrable, and $F(u,v)$ is integrable, we have that the following Fourier transform pair exists:

$$F \{ f(x,y) \} = F(u,v) = \iint_{-\infty}^{\infty} f(x,y) \exp [-j2\pi(ux + vy)] dx dy \quad (1-9)$$

and

$$F^{-1} F(u,v) = f(x,y) = \iint_{-\infty}^{\infty} F(u,v) \exp [j\pi(ux + vy)] du dv \quad (1-10)$$

where u and v are the frequency variables.

As in the one-dimensional case, the Fourier spectrum, phase, and energy spectrum are, respectively, given by the relations:

$$|F(u,v)| = [R^2(u,v) + I^2(u,v)]^{1/2} \quad (1-11)$$

$$\Phi(u,v) = \tan^{-1} \left[\frac{I(u,v)}{R(u,v)} \right] \quad (1-12)$$

$$\text{and } E(u,v) = R^2(u,v) + I^2(u,v) \quad (1-13)$$

Example: The Fourier transform of the function shown in Fig. 2(a) is given by:

$$\begin{aligned}
 F(u, v) &= \iint_{-\infty}^{\infty} f(x, y) \exp [-j2\pi(ux + vy)] dx dy \\
 &= AXY \frac{\sin(\pi uX)}{(\pi uX)} \frac{\sin(\pi vY)}{(\pi vY)} e^{-j\pi uX} e^{-j\pi vY}
 \end{aligned}$$

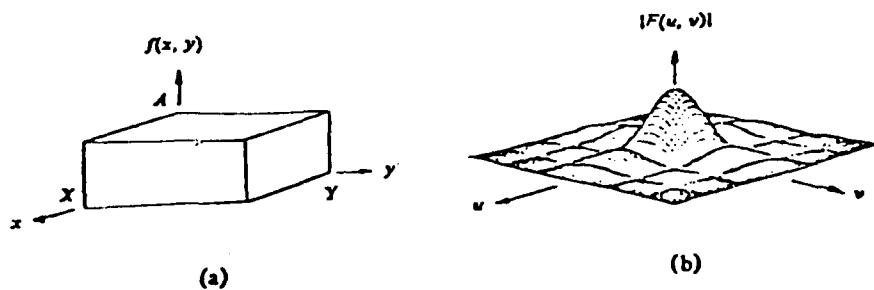


Figure 2(a).

A two-dimensional function, (b) its Fourier spectrum.

The spectrum is given by

$$\left| F(u, v) \right| = AXY \left| \frac{\sin(\pi uX)}{(\pi uX)} \right| \left| \frac{\sin(\pi vY)}{(\pi vY)} \right|$$

A plot of this is shown in Fig. 2(b) in three-dimensional perspective.

THE DISCRETE FOURIER TRANSFORM

Let us now assume that a continuous function $f(x)$ is discretized into a sequence $\{f(x_0), f(x_0 + x), f(x_0 + 2\Delta x), \dots, f(x_0 + [N - 1]\Delta x)\}$ by taking N samples Δx units apart, and that we define

$$f(x) = f(x_0 + x\Delta x) \quad (2-1)$$

where x now assumes the discrete values $0, 1, 2, \dots, N - 1$. In other words, the sequence $\{f(0), f(1), f(2), \dots, f(N - 1)\}$ will be used to denote any N uniformly spaced samples from a corresponding continuous function.

With the above notation in mind, we have that discrete Fourier transform pair that applies to sampled functions is given by

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \exp [-j2\pi ux/N] \quad (2-2)$$

for $u = 0, 1, 2, \dots, N - 1$, and

$$f(x) = \sum_{u=0}^{N-1} F(u) \exp [j2\pi ux/N] \quad (2-3)$$

for $x = 0, 1, 2, \dots, N-1$.

The values of $u = 0, 1, 2, \dots, N-1$ in the discrete Fourier transform given in Eq. (2-2) correspond to samples of the continuous transform at values $0, \Delta u, 2\Delta u, \dots, (N-1)\Delta u$. In other words we are letting $F(u)$ represent $F(u\Delta u)$. This notation is similar to that used for the discrete $f(x)$, with the exception that the samples of $F(u)$ start at the origin of the frequency axis. It can be shown that Δu and Δx are related by the expression

$$u = \frac{1}{N \Delta x} \quad (2-4)$$

For the two variable case the discrete Fourier transform pair is given by the equations

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \exp [-j2\pi(ux/M + vy/N)] \quad (2-5)$$

for $u = 0, 1, 2, \dots, M-1, v = 0, 1, 2, \dots, N-1$, and

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \exp [j2\pi(ux/M + vy/N)] \quad (2-6)$$

for $x = 0, 1, 2, \dots, N-1$ and $y = 0, 1, 2, \dots, N-1$.

Sampling of a continuous function is now in a two-dimensional grid with divisions of width Δx and Δy in the x and y axis, respectively. As in the one-dimensional case, the discrete function $f(x,y)$ represents samples of the function $f(x_0 + x\Delta x, y_0 + y\Delta y)$ for $x=0, 1, 2, \dots, M-1$ and $y = 0, 1, 2, \dots, N-1$. Similar comments hold for $F(u,v)$. The sampling increments in the spatial and frequency domains are related by

$$u = \frac{1}{M\Delta x} \quad (2-7)$$

and

$$v = \frac{1}{N\Delta y} \quad (2-8)$$

When images are sampled in a square array $M = N$ and

$$F(u,v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(xy) \exp [-j2\pi(ux + vy) / N] \quad (2.9)$$

for $u, v = 0, 1, 2, \dots, N-1$ and

$$f(x,y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v) \exp [j2\pi(ux + vy) / N] \quad (2.10)$$

for $x, y = 0, 1, 2, \dots, N-1$. Note that in the case we have included a $1/N$ term in both expressions. Since $F(u,v)$ and $f(x,y)$ are a Fourier transform pair, the grouping of these constant multiplicative terms is arbitrary.

EXAMPLE: As an illustration of the use of Eqs. (2-2) and (2-3), consider the functions shown in Fig. 5(a). If this function is sampled at the argument values $x_0 = 0.5$, $x_1 = 0.75$, $x_2 = 1.0$, $x_3 = 1.25$, and if the argument is redefined as discussed above, we obtain the discrete function shown in Fig. 3(b).

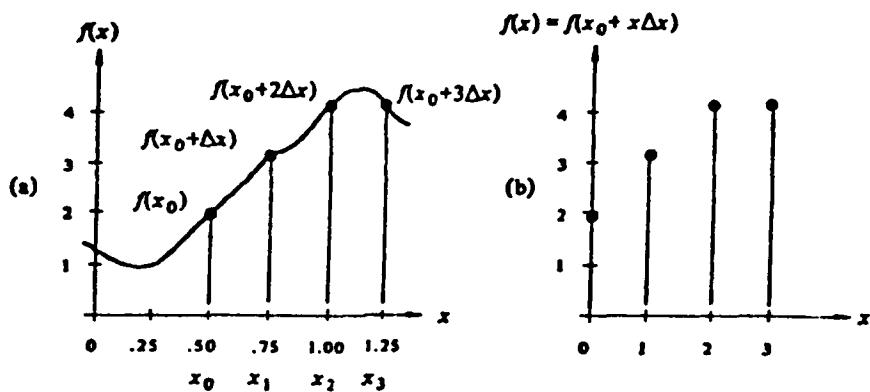


Figure 3.

A simple function and samples in the x domain. In (a) x is a continuous variable; in (b) x is discrete.

Application of Eq. (2-2) yields the following:

$$F(0) = \frac{1}{4} \sum_{x=0}^3 f(x) \exp [0]$$

$$= \frac{1}{4} [f(0) + f(1) + f(2) + f(3)] = 3.25$$

$$F(1) = \frac{1}{4} \sum_{x=0}^3 f(x) \exp [-j2\pi x/4]$$

$$= \frac{1}{4} [2e^0 + 3e^{-j\pi/2} + 4e^{-j\pi} + 4e^{-j3\pi/2}] = \frac{1}{4} [-2+j].$$

Notice that all values of $f(x)$ contribute to each of the four terms of the discrete Fourier transform. Conversely, all terms of the transform contribute in forming the inverse transform.

SOME PROPERTIES OF THE TWO-DIMENSIONAL FOURIER TRANSFORM

While our primary interest is in two-dimensional, discrete transforms, the underlying concepts of some of these properties are much easier to grasp if they are presented in their one-dimensional, continuous form.

PERIODICITY AND CONJUGATE SYMMETRY

The discrete Fourier transform and its inverse are periodic with period N ; that is,

$$F(u, v) = F(u + N, v) = F(u, v + N) = F(u + N, v + N) \quad (3.1)$$

Although Eq. (3-1) points out that $F(u, v)$ repeats itself for an infinite number of values of u and v , only the N values of each variable in any one period are required to obtain $f(x, y)$ from $F(u, v)$. In other words, only one period of the transform is necessary to completely specify $F(u, v)$ in the frequency domain. Similar comments hold for $f(x, y)$ in the spatial domain.

The Fourier transform also exhibits conjugate symmetry since

$$F(u, v) = F^*(-u, -v) \quad (3-2)$$

or more interestingly,

$$|F(u, v)| = F|(-u, -v)| \quad (3-3)$$

As mentioned earlier, it is often of interest to display the magnitude of the Fourier transform for interpretation purposes.

The same observations hold for the magnitude of the two-dimensional Fourier transforms, with the exception that the results are considerably more difficult to interpret if the origin of the transform is not shifted to the frequency point $(N/2, N/2)$.

DISTRIBUTIVITY AND SCALING

It follows directly from the definition of the continuous or discrete transform pair that

$$F \{f_1(x, y) + f_2(x, y)\} = F \{f_1(x, y)\} + F \{f_2(x, y)\} \text{ and, in general}$$

that

$$F \{f_1(x, y) \cdot f_2(x, y)\} \neq F \{f_1(x, y)\} \cdot F \{f_2(x, y)\}$$

In other words, the Fourier transform and its inverse are distributive over addition, but not over multiplication.

It is also easy to show that for two scalars a and b ,

$$af(x, y) \Leftrightarrow aF(u, v)$$

and

$$f(ax, by) \Leftrightarrow \frac{1}{|ab|} F *u/a, v/b)$$

The two-dimensional, discrete convolution is formulated by letting $f(x, y)$ and $g(x, y)$ be discrete arrays of size $A \times B$ and $C \times D$, respectively. As in the one-dimensional case, these arrays must be assumed periodic with some period M and N in the x and y directions, respectively. Wraparound error in the individual convolution periods is avoided by choosing

$$M > A + C - 1 \quad (3-9)$$

and

$$N > B + D - 1 \quad (3.10)$$

The periodic sequences are formed by extending $f(x,y)$ and $g(x,y)$ as follows:

$$f_e(x,y) = \begin{cases} f(x,y) & 0 \leq x \leq A-1 \quad \text{and} \quad 0 \leq y \leq B-1 \\ 0 & A \leq x \leq M-1 \quad \text{or} \quad B \leq y \leq N-1 \end{cases}$$

and

$$g_e(x,y) = \begin{cases} g(x,y) & 0 \leq x \leq C-1 \quad \text{and} \quad 0 \leq y \leq D-1 \\ 0 & C \leq x \leq M-1 \quad \text{or} \quad D \leq y \leq N-1 \end{cases}$$

The two-dimensional convolution of $f_e(x,y)$ and $g_e(x,y)$ is given by the relation

$$f_e(x,y) * g_e(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f_e(m,n) g_e(x-m, y-n) \quad (3.11)$$

for $x = 0, 1, 2, \dots, M-1$ and $y = 0, 1, 2, \dots, N-1$. The $M \times N$ array given by this equation is one period of the discrete, two-dimensional convolution. If M and N are chosen as indicated above this array is guaranteed to be free of interference from other adjacent periods. All computations use these extended functions $f_e(x,y)$ and $g_e(x,y)$.

AVERAGE VALUE

A widely-used definition of the average value of a two-dimensional discrete function is given by the expression

$$\bar{f}(x,y) = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) = \frac{1}{N} f(0,0) \quad (3.4)$$

CONVOLUTION and CORRELATION

We will now outline two Fourier transform relationships which constitute a fundamental link between the spatial and frequency domains. These relationships, called convolution and correlation, are of fundamental importance in developing a firm understanding of image processing techniques based on the Fourier transform.

CONVOLUTION

The convolution of two functions $f(x)$ and $g(x)$, denoted by $f(x)*g(x)$, is defined by the integral

$$f(x)*g(x) = \int_{-\infty}^{\infty} f(\alpha) g(x - \alpha) d\alpha \quad (3-5)$$

where α is the variable of integration.

Two-dimensional convolution is analogous in form to Eq. (3-5). For two functions $f(x,y)$ and $g(x,y)$, we have

$$f(x,y)*g(x,y) = \iint_{-\infty}^{\infty} f(\alpha, \beta) g(x - \alpha, y - \beta) d\alpha d\beta \quad (3-6)$$

The convolution theorem in two-dimensions is summarized by the relations

$$f(x,y)*g(x,y) \Leftrightarrow F(u,v)G(u,v) \quad (3-7)$$

and

$$f(x,y)g(x,y) \Leftrightarrow F(u,v)*G(u,v) \quad (3-8)$$

From a practical point of view, it is often more efficient to compute the discrete convolution in the frequency domain instead of using Eq. (3-11) directly. The procedure is to compute the Fourier transforms of $f_e(x, y)$ and $g_e(x, y)$ by using a fast Fourier transform (FFT) algorithm. The two transforms are then multiplied and the inverse Fourier transform of the product will yield the convolution function. A comparison by Brigham (1974) shows that, for one-dimensional arrays, the FFT approach is faster if the number of points is greater than 32. Although this figure is dependent on the particular machine and algorithms used, it is well below the number of points in a row or column of a typical image.

CORRELATION

The correlation of two continuous functions $f(x)$ and $g(x)$ denoted by $f(x) \circ g(x)$, is defined by the relation

$$f(x) \circ g(x) = \int f(\alpha) g(x + \alpha) d\alpha \quad (3-12)$$

To perform correlation one slides $g(x)$ by $f(x)$ and integrate the product from $-\infty$ to ∞ for each displacement x .

The discrete equivalent of continuous correlation is defined by

$$f_e(x) \circ g_e(x) = \sum_{m=0}^{M-1} f_e(m) g_e(x + m) \quad (3-13)$$

for $x = 0, 1, 2, \dots, M - 1$. The same comments made above with regard to $f_e(x)$ and $g_e(x)$, the assumed periodicity of these functions, and the choice of values for M , apply to this case.

Similar expressions hold for two dimensions. If $f(x, y)$ and $g(x, y)$ are functions of continuous variables, their correlation is defined as

$$f(x, y) \circ g(x, y) = \iint_{-\infty}^{\infty} f(\alpha, \beta) g(x + \alpha, y + \beta) d\alpha d\beta \quad (3-14)$$

and for the discrete case we have

$$f_e(x,y) \circ g_e(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f_e(m, n) g_e(x+m, y+n) \quad (3-15)$$

for $x = 0, 1, 2, \dots, M-1$ and $y = 0, 1, 2, \dots, N-1$. As in the case of discrete convolution, $f_e(x,y)$ and $g_e(x,y)$ are extended functions, and M and N are chosen according to avoid wraparound error in the periods of the correlation function.

For both the continuous and discrete cases the following correlation theorem holds:

$$f(x,y) \circ g(x,y) \Leftrightarrow F(u,v) G^*(u,v)$$

and

$$f(x,y) g^*(x,y) \Leftrightarrow F(u,v) \circ G(u,v)$$

where "*" denotes the complex conjugate. Note: for discrete variables, all functions are assumed to be extended and periodic.

One of the principal applications of correlation in image processing is in the area of template or prototype matching, where the problem is to bind the closest match between a given unknown image and a set of images of known origin. One approach to this problem is to compute the correlation between the unknown and each of the known images. The closest match can then be found by selecting the image that yields the correlation function with the largest value. Since the resultant correlations are two-dimensional functions, this involves searching for the largest amplitude of each function. As in the case of discrete convolution, the computation of $f(x,y) \circ g_e(x,y)$ is often more efficiently carried out in the frequency domain using an FFT algorithm to obtain the forward and inverse transforms.

THE FAST FOURIER TRANSFORM

The number of complex multiplications and additions required to implement Eq. (2-2) is proportional to N^2 . This can be seen easily by noting

that, for each of the N values of u , expansion of the summation sign requires N complex multiplications of $f(x)$ by $\exp[-j2\pi ux/N]$ can be computed once and stored in a table for all subsequent applications. For this reason, the multiplication of u by x in these terms is usually not considered a direct part of the implementation.

In this section it is shown that, by properly decomposing the number of multiply and add operations it can be made proportional to $N \log_2 N$. The decomposition procedure is called the Fast Fourier transform (FFT) algorithm. The reduction in proportionality from N^2 to $N \log_2 N$ multiply/add operations represents a significant savings in computation effort, as shown by the figures in the table. It is evident from this table that the FFT approach offers a considerable computational advantage over a direct implementation of the Fourier transform, particularly when N is relatively large. For example, a direct implementation of the transform for $N=8192$ requires on the order of three-quarters of an hour in a machine such as an IBM 7094. By contrast, the same job can be done in this machine in about 5 sec. using an FFT algorithm.

N	N^2 (Conventional FT)	$N \log_2 N$ (FFT)	Computational Advantage ($N/\log_2 N$)
2	4	2	2.00
4	16	8	2.00
8	64	24	2.67
16	256	64	4.00
32	1,024	160	6.40
64	4,096	384	10.67
128	16,384	896	18.29
256	65,536	2,048	32.00
512	262,144	4,608	56.89
1024	1,048,576	10,240	102.40
2048	4,194,304	22,528	186.18
4096	16,777,216	49,152	341.33
8192	67,108,864	106,496	630.15

As pointed out previously, a two-dimensional Fourier transform can be readily computed by a series of applications of the one-dimensional transform.

THE INVERSE FFT

Thus far, little has been said concerning the inverse Fourier transform. It turns out that any algorithm for implementing the discrete forward transform can also be used (with minor modifications in the input) to compute the inverse.

For a two-dimensional square array we take the complex conjugate, that is,

$$f^*(x,y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F^*(u,v) \exp [-j2\pi(ux + vy)/N]$$

which we see is in the form of the two-dimensional forward transform. It follows, therefore, that if we input $F^*(u,v)$ into an algorithm designed to compute the forward transform, the result will be $f^*(x,y)$. By taking the complex conjugate of this result we obtain $f(x,y)$. In the case where $f(x)$ or $f(x,y)$ are real, the complex conjugate operation is unnecessary since $f(x) = f^*(x)$ and $f(x,y) = f^*(x,y)$ for real functions.

The fact that the two-dimensional transform is usually computed by successive passes of the one-dimensional transform is a frequent source of confusion when using the above technique to obtain the inverse. When using a one-dimensional algorithm to compute the two-dimensional inverse, the method is not to compute the complex conjugate after each row or column is processed. Instead, the function $F^*(u,v)$ is treated as if it were $f(x,y)$ in the forward, two-dimensional transform procedure. The complex conjugate of the result (if necessary) will yield the proper inverse $f(x,y)$.

S E C T I O N I I I

O T H E R T R A N S F O R M T E C H N I Q U E S

26. Blank

A number of methods exist which generalize the Fourier transform either by considering the class of Fast Unitary Transforms (which include the FFT) or by considering group characters. The practical interest here is in the computational efficiency inherent in these more general transforms.

For an arbitrary sequence of functions, the Gram-Schmidt process generates a sequence of orthogonal functions. Any continuous function can be expressed via this orthogonal set with minimum L_2 error; for certain classes of functions the convergence is uniform. Most important among these are the Haar functions, which assume 2 values, and the Walsh functions with values ± 1 . Both have discrete analogies and Fast Discrete Haar/Walsh transforms are computable, FHT and FWT. Because of the simplicity of the basic functions much greater computational savings can be obtained than from FFT (up to 30 times faster than FFT).

Advantageous use can be made of FHT/FWT in certain applications; e.g. in data transmission/reconstruction in which one represents a given signal in some sense and reconstruct it from the minimal representation. A number of serious difficulties arise with these transforms due to the fact that the relation with the circle has been lost. For example: 1) no natural interpretation in terms of frequency exists (the "sequency" viewpoint of Harmuth for Walsh functions lacks physical meaning); 2) due to absence of the circle relationship the important convolution theorem is not available (forced analogies to a convolution theorem via dyadic convolutions have been made but their interpretations are not clear).

The striking advantages of FWT and FHT over the usual FFT in computational effort should motivate further investigation in this area.

The historical situation regarding orthogonal functions at the beginning of the twentieth century was one of well known and useful kinds of such functions: the trigonometric functions which occur in Fourier series; orthogonal polynomials such as those of Legendre, Hermite, and Laguerre;

Bessel's functions, the Sturm-Liouville series, and other special functions. But, there was no general theory embracing all such systems of functions.

The Hungarian mathematician, Alfred Haar, was concerned with convergence properties of series of orthogonal functions, and also constructing a new set (now called the Haar system) of such functions. He defined a set of orthogonal functions each taking essentially only two values such that the formal expansion of an arbitrary continuous function in those functions converges uniformly to the given function, a property not possessed by orthogonal sets known up to that time.

.. In 1923, J.L. Walsh published a set of orthogonal functions which are complete on the interval [0,1]; they take only the values ± 1 , and are similar in oscillation and many other properties to the trigonometric functions. They have turned out to have important practical applications in calculation.

The limits of the usefulness of these functions both in theoretical work and in engineering applications still seem to be undetermined.

Traditionally, the theory of communication has been based on the complete, orthogonal system of sine and cosine functions. The concept of frequency is defined as the parameter f in $\sin 2\pi f t$ and $\cos 2\pi f t$. The question arises whether there are other systems of functions on which theories of similar scope can be based, and that lead to equipment of practical interest.

The parameter in $\sqrt{2}\sin 2i\pi\theta$ and $\sqrt{2}\cos 2i\pi\theta$ gives the number of oscillations in the interval $-1/2 \leq \theta < 1/2$ (that is, the normalized frequency $i=ft$). One may interpret i as "one half the number of zero crossings per unit time" rather than as "oscillations per unit time". (The zero crossing at the left side, $\theta = -1/2$, but not the one at the right side, $\theta = +1/2$, of the time interval is counted for sine functions).

The parameter i also equals one half the number of zero crossings in the interval $-1/2 \leq \theta \leq 1/2$ for Walsh functions. In contrast to sine-cosine functions, the sign changes are not equidistant. If i is not an integer, then it equals "one half the average number of zero crossings per unit time". The term "normalized sequency" has been introduced for L and $\Phi = i/T$ is called the normalized sequency. Sequency in zps = $1/2$ (average number of zero crossings per second).

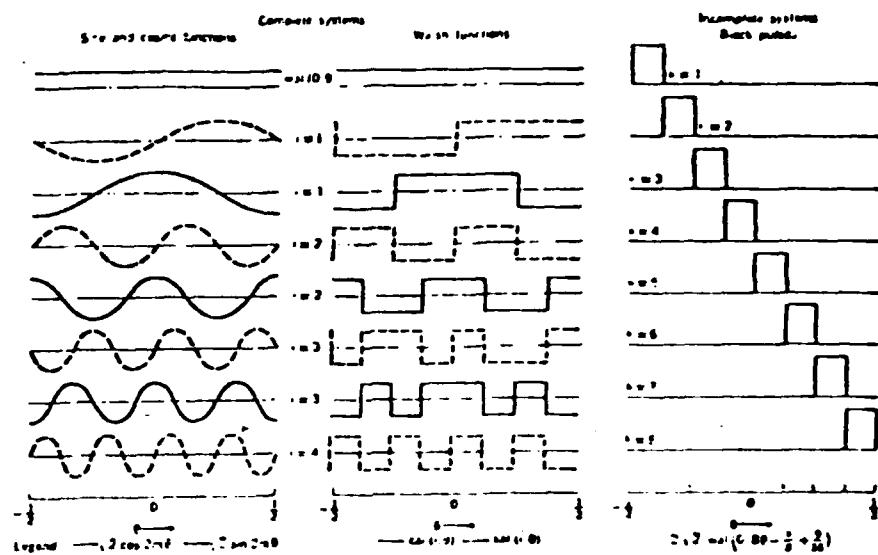
The general form of a sine function $V \sin (2\pi ft + \alpha)$ contains the parameters amplitude V , frequency f , and phase angle α . The general form of a Walsh function $V \text{ sal} (\Phi T, t/T + t_o/T)$ contains the parameters amplitude, V , sequency, Φ , the delay, t_o , and time base, T . The normalized delay, t_o/T , corresponds to the phase angle. The time base, T , is an additional parameter and it causes a major part of the differences in the applications of sine-cosine and Walsh functions.

So far, Walsh functions are the only known functions with desirable features comparable to sine-cosine functions for use in communications. Development of semi-conductor technology has imparted practical interest in them at this time. Generally speaking, the transition from sine-cosine functions to other complete systems means a transition from linear, time-invariant components and equipment to linear, time-variable components and equipment, which, of course, constitute a much larger class. The mathematical theory of Walsh-Fourier analysis corresponds to the Fourier analysis used for sine-cosine functions. There is no theory of similar scope for block pulses, because they are incomplete.

The sal and cal transforms of Walsh-Fourier analysis are defined by

$$\begin{aligned} a_s(\mu) &= \int_{-\infty}^{\infty} F(\theta) \text{ sal}(\mu, \theta) d\theta \\ a_c(\mu) &= \int_{-\infty}^{\infty} F(\theta) \text{ cal}(\mu, \theta) d\theta \end{aligned}$$

a) Orthonormal systems of functions.



b) List of features and applications of sine-cosine functions, Walsh functions, and block

	Sine Functions	Walsh Functions	Block Pulses
Parameters	Amplitude Frequency Phase angle ...	Amplitude Seqency Delay Time base	Amplitude ... Pulse position Pulse width
Mathematical theory	Fourier analysis	Walsh-Fourier analysis	...
Power spectrum	Frequency spectrum	Seqency spectrum	...
Filters	Time invariant, linear	Periodically time variable, linear	Time variable, linear
Characterization	Frequency response of attenuation and phase shift	Seqency response of attenuation and delay	Attenuation as function of time
Multiples	Frequency division	Seqency division	Time division
Modulation	Amplitude, phase, frequency modulation	Amplitude, time position, time base, code modulation	Amplitude, pulse position, pulse width modulation
Radiable	$\sin 2\pi f t$, $\cos 2\pi f t$	$\text{sal}\left(\mu, \frac{t}{T}\right)$, $\text{cal}\left(\mu, \frac{t}{T}\right)$...

$$F(\theta) = \int_{-\infty}^{\infty} [\alpha_s(\mu) \text{ sal}(\mu \theta) + \alpha_c(\mu) \text{ cal}(\mu \theta)] d\mu$$

Walsh-Function Filters - For a sequency low pass filter based on Walsh functions the input signal, $F(\theta)$, is transformed into a step function. $F_{\text{ff}}(\theta)$, with steps of a certain width, by integrating $F(\theta)$ during an interval equal to the step width. The amplitudes of the steps are chosen so that $F_{\text{ff}}(\theta)$ yields a least-mean-square approximation of $F(\theta)$. In addition, $F_{\text{ff}}(\theta)$ is delayed with respect to $F(\theta)$ by one step width.

The number of samples obtained is equal to twice the cut-off sequency. Hence, the sampling theorems of Fourier analysis permit the comparison of frequency and sequency filters.

Theorems for the multiplications of Walsh functions have been proven. These are:

$$\begin{aligned} \text{cal}(k, \theta) \text{ cal}(i, \theta) &= \text{cal}[k \oplus i, \theta] \\ \text{sal}(k, \theta) \text{ cal}(i, \theta) &= \text{sal}[i \oplus (k-1) + 1, \theta] \\ \text{sal}(k, \theta) \text{ sal}(i, \theta) &= \text{cal}[(k-1) \oplus (i-1), \theta] \end{aligned}$$

where the symbol \oplus indicates modulo 2 addition. Note that the product of two Walsh functions yields only one Walsh function. Therefore, the amplitude modulation of a Walsh carrier yields only one sequency sideband as compared to the two sidebands obtained when a sine carrier is modulated. A typical application of the multiplication theorems of Walsh functions is in the design of sequency-bandpass filters.

Digital Filtering and Multiplexing - One of the most promising aspects of Walsh functions is the ease with which filters and multiplex equipment can be implemented as digital circuits. The reason is that numerical Walsh-Fourier transformation and numerical sequency shifting of signals require summations and subtractions only. In the case of sine-cosine functions, the corresponding operations require multiplications with irrational numbers.

A digital filter based on Walsh functions can be readily obtained. The input signal passes first through a sequency low-pass filter then transforms it into a step function. This step function is sampled and the samples are transformed into numbers by an analog/digital converter. A series of these numbers is stored in a digital storage. A Walsh-Fourier transform of this series is obtained by performing certain additions and subtractions in an arithmetic unit. Some or all of the obtained coefficients, that represent sequency components, may be suppressed or altered - in effect, a filtering process. An inverse Walsh-Fourier transform yields the filtered signal as a series signal by digital/analog converter. Since there is a fast Walsh-Fourier transform just as there is a fast Fourier transform, the arithmetic operations in a digital sequency filter are not only simpler than in a digital frequency but can be performed faster.

One of the features of Walsh functions that makes them of some interest in signal processing is the fact that their amplitudes are given precisely by a single bit, so that their use does not directly contribute to roundoff noise. The basis vectors of symmetry analysis offer the same attraction with, additionally, for low orders of input data frames N , some economy of computations by reason of the zeros.

OTHER IMAGE TRANSFORMS

The Fourier transform is the transform most often used in image processing applications; there are other transforms which are also of interest in this area.

The one-dimensional, discrete Fourier transform is one of a class of important transforms which can be expressed in terms of the general relation

$$T(u) = \sum_{x=0}^{N-1} f(x)g(x,u) \quad (5-1)$$

where $T(u)$ is the transform of $f(x)$, $g(x,u)$ is the forward transformation kernel, and u assumes values in the range 0, 1, ..., $N-1$. Similarly, the inverse transform is given by the relation

$$f(x) = \sum_{u=0}^{N-1} T(u)h(x,u) \quad (5-2)$$

where $h(x,u)$ is the inverse transformation kernel and x assumes values in the ranges 0, 1, ..., $N-1$. The nature of a transform is determined by the properties of its transformation kernel.

For two dimensional square arrays the forward and inverse transforms are given by the equations

$$T(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y)g(x,y,u,v) \quad (5-3)$$

and

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u,v)h(x,y,u,v) \quad (5-4)$$

where, as above, $g(x,y,u,v)$ and $h(x,y,u,v)$ are called the forward and inverse transformation kernels, respectively.

The two dimensional Fourier transform has the kernel

$$g(x,y,u,v) = \frac{1}{N} \exp \left[-j2\pi (ux + vy) / N \right]$$

which is separable and symmetric since

$$\begin{aligned} g(x,y,u,v) &= g_1(x,u)g_1(y,v) \\ &= \frac{1}{\sqrt{N}} \exp \left[-j2\pi ux/N \right] \frac{1}{\sqrt{N}} \exp \left[-j2\pi vy/N \right] \end{aligned}$$

It is easily shown that the inverse Fourier kernel is also separable and symmetric.

A transform with a separable kernel can be computed in two steps, each requiring a one dimensional transform. First, the one dimensional transform is taken along each row of $f(x,y)$, yielding

$$T(x,v) = \sum_{y=0}^{N-1} f(x,y)g_2(y,v) \quad (5-5)$$

for $x, v = 0, 1, 2, \dots, N-1$. Next, the one dimensional transform is taken along each column of $T(x,v)$; this results in the expression

$$T(u,v) = \sum_{x=0}^{N-1} T(x,v)g_1(x,u) \quad (5-6)$$

for $u, v = 0, 1, 2, \dots, N-1$. The same final results are obtained if the transform is taken first along each column of $f(x,y)$ to obtain $T(y,u)$ and then along each row of the latter function to obtain $T(u,v)$. Similar comments hold for the inverse transform if $h(x,y,u,v)$ is separable.

If the kernel $g(x,y,u,v)$ is separable and symmetric, Eq. (5-3) can also be expressed in the following matrix form:

$$T = AFA \quad (5-7)$$

where F is the $N \times N$ image matrix, A is an $N \times N$ symmetric transformation matrix with elements $a_{ij} = g_1(i,j)$, and T is the resulting $N \times N$ transform for values of u and v in the range, $0, 1, 2, \dots, N-1$.

To obtain the inverse transform we pre-multiply and post-multiply Eq. (5-7) by an inverse transformation matrix B .

If $B = A^{-1}$, it then follows that

$$F = BTB$$

which indicates that the digital image F can be recovered completely from its transform. If B is not equal to A^{-1} , then we obtain an approximation to F , given by the relation

$$F = BAFAB$$

A number of transforms, including the Fourier, Walsh, and Haar transforms, can be expressed in this form. An important property of the resulting transformation matrices is that they can be decomposed into products of matrices with fewer non-zero entries than the original matrix. This result, first formulated by Good (1958) for the Fourier transform, reduces redundancy and, consequently, the number of operations required to implement a two-dimensional transform. The degree of reduction is equivalent to that achieved by an FFT algorithm, being on the order of $N \log_2 N$ multiply/add operations for each row or column of an $N \times N$ image.

Walsh Transform

When $N = 2^n$, the discrete Walsh transform of a function $f(x)$, denoted by $W(u)$, is obtained by substituting the kernel

$$g(x, u) = \frac{1}{N} \prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)} \quad (5-8)$$

into Eq. (5-1). In other words,

$$W(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \prod_{i=0}^{N-1} (-1)^{b_i(x)b_{n-1-i}(u)} \quad (5-9)$$

where $b_k(z)$ is the k th bit in the binary representation of z . For example, if $n = 3$ and $z = 6$ (110 in binary), we have that $b_0(z) = 0$, $b_1(z) = 1$, and $b_2(z) = 1$.

The values of $g(x,u)$, excluding the $1/N$ constant term, are listed below for $N = 8$. The array formed by the Walsh transformation kernel is

Values of the Walsh transformation kernel for $N = 8$.

$u \backslash x$	0	1	2	3	4	5	6	7
0	+	+	+	+	+	+	+	+
1	+	+	+	+	-	-	-	-
2	+	+	-	-	+	+	-	-
3	+	+	-	-	-	-	+	+
4	+	-	+	-	+	-	+	-
5	+	-	+	-	-	+	-	+
6	+	-	-	+	+	-	-	+
7	+	-	-	+	-	+	+	-

a symmetric matrix whose rows and columns are orthogonal. These properties, which hold in general, lead to an inverse kernel which is identical to the forward kernel, except for a constant multiplicative factor of $1/N$. Thus, the inverse Walsh transform is given by

$$f(x) = \sum_{u=0}^{N-1} W(u) \prod_{i=0}^{n-1} (-1)^{b_1(x)b_{n-1-i}(u)} \quad (5-10)$$

Notice that, unlike the Fourier transform which is based on trigonometric terms, the Walsh transform consists of a series expansion of basis functions whose values are either plus or minus one.

It is also of interest to note that the forward and inverse Walsh transforms differ only by the $1/N$ term. Thus, any algorithm for computing the forward transform can be used directly to obtain the inverse transform simply by multiplying the result of the algorithm by N .

The forward and inverse Walsh transforms are also equal given by

$$W(u,v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u) + b_i(y)b_{n-1-i}(v)} \quad (5-11)$$

and

$$f(x,y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} W(u,v) \prod_{i=0}^{n-1} (-1)^{v_i(x)b_{n-1-i}(u) + b_i(y)b_{n-1-i}(v)} \quad (5-12)$$

Thus, any algorithm which is used to compute the two dimensional forward Walsh transform can also be used without modification to compute the inverse transform.

The Walsh transform can be computed by a fast algorithm identical in form to the successive doubling method for the FFT. The only difference is that all exponential terms W_N are set equal to one in the case of the fast Walsh transform (FWT).

The Walsh transform is real, thus requiring less computer storage for a given problem than the Fourier transform, which is in general complex valued.

S E C T I O N I V

R E F E R E N C E S

40. Blank

REFERENCES:

Arking, Lo and Rosenfeld: "A Fourier Approach to Cloud Motion Estimation," Journal of Applied Meteorology, Volume 17, pp 375-744.

Bracewell R. The Fourier Transform and its Applications, 2nd Edition, McGraw-Hill, 1974.

Childers and Durling, Digital Filtering and Signal Processing, West Publications, 1975.

Leese, Novak and Clark, "An Automated Technique for Obtaining Cloud Motion from Geosynchronous Satellite Data using Cross Correlation," Journal of Applied Meteorology, Volume 10, 1971, pp. 118-132.

Leese and Epstein, "Application of Two-Dimensional Spectral Analysis to the Quantification of Satellite Cloud Photographs," Journal of Applied Meteorology, Volume 2, 1963, pp. 629-644.

Lo, Mohr and Parikh, "Applications of Fourier Transform Methods of Cloud Movement Estimation to Simulated and Satellite Photographs, N. of Maryland." Computer Science Center Technical Report, TR-292, January, 1974.

Oppenheim and Schafer, Digital Signal Processing, Prentice Hall, 1975.

Rosenfeld and Kak, Digital Picture Processing, Academic Press, 1976.

Schreiber and Ferell, Ed., "Applications of Walsh Functions and Sequency Theory," IEEE, 1974.

SECTION V

PROGRAM DESCRIPTION AND USE

44- Blank

INTRODUCTION

The program which controls all of the calculations and output related to this task is called WETCOR. It is structured in modular form and is based on a keyword input format. This arrangement facilitates making a large number of runs using different data input, different output formats and a variety of computations while requiring simple keyword card changes.

The program reads data from a permanent file, sets up arrays of specified size and computes the auto correlation or cross correlation matrix for the two arrays via a very efficient two dimensional Fast Fourier Transform (FFT) code. The algorithm was between 5 and 20 times faster than other FFT implementations on the benchmark cases we ran. Output from the use of a two dimensional square wave test case was fully consistent with theoretical analysis. This test was used because of its simple structure, and well known properties while at the same time providing a good check on the accuracy and correctness of the code.

During the development of the WETCOR package parallel efforts considered alternatives to the FFT approach. Much attention was given to methods which offered savings in processing time. Contemporary journals on information and image processing contain an abundance of references to Walsh, Haar and other orthogonal families of functions and their corresponding "fast transforms." The greatest strength of both the Haar and Walsh transforms is the reduction in processing times by factors of 2 to 20 over comparable times using the FFT algorithm. However, as related to this specific problem they are not very useful, at the present time. The principal drawback is the difficulty in interpreting and/or decoding the convolution/deconvolution equations which are used in the discrete fast Haar and fast Walsh transforms.

WETCOR does correlation analyses of data arrays which are based on raster scans of weather patterns. The input data are stored in bytes, each containing the pixels from a raster. A portion of an input raster is selected as a base array. Another array is selected and correlated against the

46 - Blank

base array using the Fast Fourier Transform.

The program prints the initial data and the final correlation array in a compact exponential form. Other options for printing intermediate results and for handling the data differently are within the program's capabilities.

The program structure is modular. Additions and modifications are easily made.

The main structure of WETCOR is modular with the choice of the order of execution of the modules controlled by the order of the input cards. Most of the modules reset parameters. One of the main modules reads in and sets up the base array. The other reads in the second array and does the correlation using the Fast Fourier Transform.

The size of the array extracted from the file WETDAT is variable. This array is defined by its time and the position of its corner. If its size extends beyond the array stored on WETDAT, zeros are filled in. This array is then packed into an array whose dimensions must be powers of 2, which is a restriction imposed by the Fast Fourier Transform routines used. The average is subtracted from the array and the edges are rounded to avoid aliasing effects.

Both arrays are Fourier transformed; the transforms combined; and the reverse transform performed. This final array is printed. It may also be searched for the maximum and a scaled output produced which emphasizes the maximum. A centered output is available. The final array may be output on a special file.

INPUT

A - Punched Card

The input to WETCOR is free field keyword input. Each card consists of a keyword followed by any number of fields. The keyword and fields are separated by break characters: either "(" or ",". The scanning of each card is terminated by either ")", the end of the card, or "." followed by 20 blanks. The remainder of the card can be used for comments. (A card with C in the first column is treated as a comment card which will appear in the output).

The basic deck consists of:

1st	-	title card - 80 columns used for labelling.
2nd - Nth	-	keyword cards - (control the type and amount of processing to be done).
N + 1st	-	End of 7/8/9 - (terminates the run).

The keywords are described below. The fields are indicated as I for integer, R for real, and H for hollerith or character strings. An integer has only a sign and an integer string. A real has a "." and/or "E". A field containing any other character will be interpreted as a character string. If an input field has no characters or only blanks, it is treated as not being given and default values are assigned. For numerical data this usually means the value zero is used. The default values given below are the values used if that field is not used or the values used if that keyword is not used to set the values. The integers in parentheses indicate the field used; e.g. for the first case given: BASE (,7311,2261) does not use the first field; column and line numbers are given in the next 2 fields; the remaining fields are not used.

The keywords available can be divided into four basic groups:

- 1) Keywords which must be used in all runs.
- 2) Keywords which control/identify the structure of the data being used.
- 3) Keywords which identify the operations to be performed.
- 4) Keywords which control the output (format and amount of information).

1. Keyword cards required for every run: BASE, GO

(a) BASE (A1, A2, A3, A4, A5, A6, A7, A8) - the Base Keyword reads in the first base array and adjusts it to specifications.

A1 - integer - time in form hhmm. If A1 is not specified the next data file is used.

If A1 is less than -2400, data is generated for testing (see below).

A2 - integer - column number }
A3 - integer - line number } of the upper left hand corner
of the picture

A4 - integer - data value to be used (default: 1000)

A5 - integer } - lower limit - x axis (to be set)

A6 - integer } - upper limit - x axis (to be set)

A7 - integer } - lower limit - y axis (to be set)

A8 - integer } - upper limit - y axis (to be set)

(b) GO (A1, A2, A3, A4, A5) - The GO keyword reads in the second array, adjusts it and does correlation calculations.

A1 - integer - time in form hhmm. If it is not given, the next data file is used.

A2 - integer - column number

A3 - integer - line number

A4 - integer - column shift (if desired)

A5 - integer - line shift (if desired)

The shifts are applied to the data portions on WETDAT when new data is requested. The shifts are applied to the stored data when autocorrelation is being calculated.

2. Keyword cards which control/identify data structure.

(a) EVERY (A1, A2, A3, A4) - indicates column and line sampling rate on WETDAT.

A1 - integer - column rate	of array to be correlated (Go array) (default: use old values) of base array (default if A3, or A4 is blank: use values for other array. Default if the card is terminated early; use old values)
A2 - integer - line rate	
A3 - integer - column rate	
A4 - integer - line rate	

(b) REWIND

rewinds the file WETDAT.
Searching is only done in forward direction; if an earlier file is to be used a rewind is necessary before searching.

(c) INDIM (A1, A2, A3)

sets input array dimensions. This is not restricted to powers of 2 but will usually be 1/2 of DIM value.

A1 - integer -	{ set to 1 for GO array set to 2 for BASE array column dimension line dimension (default: column dimension). The defaults at load time are all 32.
A2 - integer	
A3 - integer	

(d) DIM (A1, A2, A3)

sets dimensions of working arrays. These must be powers of 2. The card fields are identical to those of INDIM. The defaults at load time are all 64.

(e) PRP (A1, A2, A3, A4, A5, A6, A7, A8) - sets the parameters controlling the routine which prepares the input arrays for the Fast Fourier transformation. Empty fields are treated as zeros. The load time defaults for both arrays are given in parentheses at the right.

A1 - integer - chooses array as with INDIM

A2 - integer - { set to 0: center array
{ set to 1: use shift values in 5 and 6
default: 1

A3 - integer - edge treatment. If several choices are wanted, sum the corresponding values. The edges of the input array are scaled by \cos^2 . This edge rounding can be eliminated by defining the edge size to be 0. The edge treatment parameter values are:

- 1: Use values 7 to define edge size in x-direction instead of $\frac{1}{16}$ dimension.
- 2: Use value 8 to define edge size in y-direction instead of $\frac{1}{16}$ dimension.
- 4: Zero edge of output, not the edge of the minimal array.
- 8: Do not propagate the edges of the input array (working array set to zero outside input range).
- 16: Set input array edges equal to input average.

default: 16

A4 - integer - Normalization parameter

- 0: Subtract average and divide by average
- 1: Divide by average; do not subtract average
- 2: Subtract average; do not divide by average
- 3: Do not normalize

default: 2

A5 - integer - z-shift Used if field 2 is non zero

A6 - integer - y-shift

A7 - integer - x edge size Used if field 3 requires edge sizes.
A8 - integer - y edge size
default: + for A5 through A8: 0.

3. Keyword Cards which identify the operations to be performed.

(a) PEAK | control the search for a peak in the correlation
 | NO PEAK | array (default at load time: NOPEAK)

(b) AUTO | control the generation of the second array from the
 | base array
 | NAUTO | for autocorrelation testing (default at load time:
 | NAUTO).

4. Keyword Cards controlling output

(a) PRINTH (V1, V2, ..., Vn) - controls which arrays are to be printed. Each field is an integer indicating a particular array to be printed during the calculation. The default is (12, 11, 6). The acceptable values for V_j and the corresponding output are as follows:

V_j = 1: Working version of GO array
 2: Working version of Base array (printed for each calculation)
 3: Transform of array to be correlated
 4: Transform of base array
 5: Direct product of transforms
 6: Inverse transform of product; (correlation array)
 7: Correlation array shifted so that origin is centered
 11: GO array to be correlated as read from WETDAT
 12: Base array as read from WETDAT
 13: Working version of base array when generated

The output arrays are in a compressed format. Each array entry is multiplied by the scaling factor printed at the top of the array. The three most significant digits greater than 1 are then output with a leading symbol indicating the sign and the power of 10 which has been suppressed.

Symbols for	positive	blank	+	*	P	P ***
	negative	-	=	X	M	M ***
	10 **	0	1	2	3	4

- (b) PRINTB (A1) Prints prepared base array
A1 REAL Scale factor to be used. (Default is 1.)
- (c) SCALE (A1, A2) Sets output scale factors. These are adjusted during calculation.
A1 - REAL - Scale for GO array to be correlated
A2 - REAL - Scale for BASE array. (Default: same scale for both).
The load time default for both is 1.
- (d) TITLE Causes the next card to be read and used as a new title card.
- (e) PRINT | Causes the keyword cards to be printed or not printed.
NOPRINT | Default is PRINT

WETDAT is read by using unformatted READ statements. The file is structured into records containing integer values as follows:

Record	
1	hour and initial column number
2-92	line number, data values for that line

Ninety-one lines are expected - each of which is expected to have at least 90 values. These numbers are in common /WETDAT/. The column and line numbers are expected to change in steps of 1 and 2 respectively, unless changed by the EVERY keyword. This could be made dependent on the more recent WETDAT files which have the steps and other information in a longer first record.

CORARY is written by unformatted WRITE statements. Each array is a file containing 3 records. The first record consists of 120 characters containing the data and time when the file is written. The second record contains 16 quantities: the array dimensions, the sampling rates, the operative keyword (probably GO), and an array of 11 items containing the input fields. The third record is the data array.

The normal output from WETCOR consists of the keyword cards which control the run and the arrays specified by those keywords. The arrays are given in the abbreviated numerical form described under PRINT H. The correlation array may be output on CORARY.

A number of methods exist which generalize the Fourier transform either by considering the class of Fast Unitary Transforms (which include the FFT) or by considering group characters. The practical interest here is in the computational efficiency inherent in these more general isforms.

For an arbitrary sequence of functions, the Gram-Schmidt process generates a sequence of orthogonal functions. Any continuous function can be expressed via this orthogonal set with minimum L_2 error; for certain classes of functions the convergence is uniform. Most important among these are the Haar functions, which assume 2 values, and the Walsh functions with values ± 1 . Both have discrete analogies and Fast Discrete Haar/Walsh transforms are computable, FHT and FWT. Because of the simplicity of the basis functions much greater computational savings can be obtained than from

FFT (up to 30 times faster than FFT).

Advantageous use can be made of FHT/FWT in certain applications: date transmission/reconstruction in which one represents a given signal in some sense and reconstruct it from the minimal representation. A number of serious difficulties arise with these transforms due to the fact that the relation with the circle has been lost. For example: 1) no natural interpretation in terms of frequency exists (the "sequency" viewpoint of Harmuth for Walsh functions lacks physical meaning); 2) due to absence of the circle relationship the important convolution theorem is not available (forced analogies to a convolution theorem via dyadic convolutions have been made but their interpretations are not clear).

The striking advantages of FWT and FHT over the usual FFT in computational effort should motivate further investigation in this area.

The historical situation regarding orthogonal functions at the beginning of the twentieth century was one of well known and useful kinds of such function: the trigonometric functions which occur in Fourier series; orthogonal polynomials such as those of Legendre, Hermite, and Laguerre; Bessel's functions, the Sturm-Liouville series, and other special functions. But, there was no general theory embracing all such systems of functions.